ROYAL HOLLOWAY, UNIVERSITY OF LONDON
BSc EXAMINATION 2021

CS1812: Object oriented programming II
CS1812R: Object oriented programming II – FOR FIRST SITS/RESIT CANDIDATES

Time allowed: **TWO hours**

Please answer **ALL** questions.

- Handwrite your answers on paper, and write your candidate number and the module number at the top of each page. Photograph/scan the pages and keep the original paper versions, as they may be required by the examiners.
- For each question you attempt, please clearly state the question number.
- Please DO NOT include your name or Student ID anywhere on your work.
- **Academic Misconduct:** We will check all assignments for academic misconduct. Suspected offences will be dealt with under the College's formal Academic Misconduct procedures. Please remember:
  - The work submitted is expected to be your own work and only your work. You may not ask for help from any source, or copy anyone else's work.
  - You must not give help to anyone else, including sending them any parts of the questions or copies of your solutions.
  - You must not discuss the questions or solutions with anyone else.
- **Submitting your work:**
  - Your document must be submitted through Moodle using the submission link in the module Moodle page. If possible please convert your document into a PDF document to make the submission process quicker and easier.
  - Emailed submissions will not be accepted.
  - **You must complete your exam upload within 1 hour of the exam finish time.**

1. This question is about writing classes and enums to represent particles in a physics simulation.

   (a) Write an enumerated type `Spin`, that contains four elements named `ZERO`, `ONE_HALF`, `ONE`, and `THREE_HALVES`. [5 marks]

   (b) Write a class `Particle`. The class should have the following.

   - Private fields: `name`, of type `String`; `mass`, of type `double`; `charge`, of type `int`; and `spin`, of the enumerated type `Spin`.
   - Public getters for each of the above fields.
   - A public constructor, that takes parameters of type `String`, `double`, `int`, and `Spin`, and uses these to initialise the fields.

   [10 marks]

   (c) Add a public instance method `equals(Object o)` to the `Particle` class that takes an object as an argument and returns a `boolean` indicating whether that object represents the same subatomic particle as the receiver. This method should check that:

   - the object passed as an argument is an instance of the `Particle` class;
   - the values of the `mass`, `charge`, and `spin` fields in the object passed as an argument are the same as those in the receiver object.

   [5 marks]

2. Consider the following two classes `MyClass` and `AnotherClass`, defined in different files:

```java
package rhul.mypackage;

import java.util.HashMap;

public class MyClass {
    int x;
    private HashMap<String,Integer> y;
    protected double z;
}
```

```java
import rhul.mypackage.*;

public class AnotherClass extends MyClass {
    int w;

    public AnotherClass(int w) {
        this.w = w;
    }

    public void foo(int w) {
        if (this.w > 0) {
            int t = 2;
            t = w + t;
        }
        else {
            int t = -2;
            t = w * t;
        }

        System.out.println("The value of t is: " + t);
    }

}
```

(a) Which packages are `MyClass` and `AnotherClass` in? [4 marks]

(b) Which instance variables of `MyClass` and `AnotherClass` are visible in the method `foo` of the class `AnotherClass`? Justify your answer. [5 marks]

(c) Compiling `AnotherClass` will indicate that there is an error in the method `foo`.

i. Explain what is causing the error. [3 marks]

ii. Modify the method `foo` so that it compiles correctly and the code

```
AnotherClass a = new AnotherClass(-3);
a.foo(4);
```

prints out "`The value of t is -8`". [3 marks]

3. Consider the following class.

```
class VendingMachine {
    protected int stockLevel;
    public VendingMachine(int initialLevel) {
        stockLevel = initialLevel;
    }
    public void dispense() {
        if (retrieve()) {
            System.out.println("Enjoy your drink!");
        } else { System.out.println("Nothing left!"); }
    }
    protected boolean retrieve() {
        if (stockLevel == 0) { return false; }
        stockLevel--;
        return true;
    }
}
```

(a) Write a class `SmartVendingMachine` that inherits from the `VendingMachine` above, for modelling a vending machine that can automatically refill itself. It should contain the following.

- A constructor that takes an `int` as a parameter and calls the superclass constructor to initialise the `stockLevel` field.
- A method overriding the `retrieve()` method from `VendingMachine` that: first checks the current stock level of the receiver and, if it is `1`, resets it to `5`; then returns the result of calling the `retrieve()` method from `VendingMachine`. Remember to use any appropriate @ annotations.

[8 marks]

NEXT PAGE

(b) Consider the following incomplete `main` method.

```
public static void main(String[] args) {
    VendingMachine m = /* missing code */
    for (int i = 0; i < 10; i++) {
        m.dispense();
    }
}
```

i. Complete the missing code to create an appropriate object for modelling a vending machine with an initial stock level of `5`, such that the result of executing the `main` method is to print out the text "`Enjoy your drink!`" ten times. [4 marks]

ii. What is the sequence of constructors from the `VendingMachine` and `SmartVendingMachine` classes that are called when the code you have given for the previous part is executed? [2 marks]

iii. What is the sequence of methods from the `VendingMachine` class and the `SmartVendingMachine` class that are called the first time the body of the `for` loop is executed? (Ensure that you write out the full name of each method, including the name of the class it is declared in). [6 marks]

4. This question is about exception handling. Consider the following incomplete method.

```
public static String getLine(int n, String fileName) {
    BufferedReader br = null;
    try {
        br = new BufferedReader(new FileReader(fileName));
    }
    catch(FileNotFoundException ex) {
        // catch block
    }
    finally {
        // finally block
    }
}
```

(a) Complete the method above so that it returns the `n`th line of the file given in `fileName` (for `n` = `0` it should return the first line). The method should return `null` if no such line can be found, or if an error has occurred whilst reading the file.

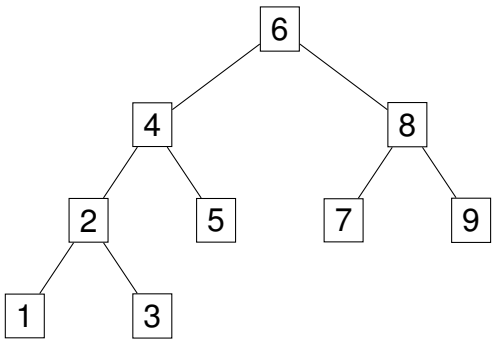
Use `readLine()` from the `BufferedReader` class to read each line of the file; this method returns the next line, or `null` if there are no lines left to read.

Reading the file may raise an `IOException`. Make sure that this is properly handled and that the `BufferedReader` is closed correctly. [11 marks]

(b) Write a class `LineNotFoundException` defining a custom exception. Indicate how you can modify your solution to part (a) so that `getLine` throws an instance of that exception when the file has fewer than `n+1` lines. [4 marks]

NEXT PAGE

5. Consider the tree shown in the diagram below.



(a) Write out the sequence of values that would be obtained by the following traversals of the tree given above.

i. Pre-order depth-first traversal [4 marks]

ii. Breadth-first traversal [4 marks]

(b) Draw a single binary tree in which nodes contain one of the characters ‘B’, ‘O’, ‘S’, or ‘T’, such that an in-order depth-first traversal of the tree spells the word “BOOST” and a post-order depth-first traversal of the same tree spells the word “BOOTS”. [7 marks]

6. The following `binarySearch` method searches a sorted array of integers `arr` for a given integer `val`, looking between between positions `start` and `end` (inclusive).

```
boolean binarySearch(int[] arr, int val, int start, int end)
{
    if(start > end) {
        return false;
    }

    int middle = (start + end) / 2;

    if(arr[middle] == val) {
        return true;
    }
    else if (val < arr[middle]) {
        return binarySearch(arr, val, start, middle - 1);
    }
    else {
        return binarySearch(arr, val, middle + 1, end);
    }
}
```

   (a) Modify this method so that it can be used to search a generic list of type `ArrayList<T>`. Use the method `get(int index)` of `ArrayList` to retrieve the element of the list at position `index`. Add comments to the lines you needed to change.
   (**Hint:** The signature should require that the type parameter `T` implements the interface `Comparable<T>`. You can use the `compareTo` method from the `Comparable` interface to compare elements. Recall that `x.compareTo(y)` returns a negative integer, zero, or a positive integer if `x` is less than, equal, or greater than `y`, respectively.) [6 marks]

   (b) Write a static generic method `printSearch`, with a type parameter `T`, that takes as input two arguments `ArrayList<T> lst` and `T val`, and does not return any value. The method should do the following:

   - check if the list is sorted in ascending order and, if it is not, print out an error message;

- if the list is sorted, call the `binarySearch` method you have implemented for part (a) to search for the element `val` in the list `lst`, and prints out a message describing the search result.

You can use the `size()` method of `ArrayList` to get the number of elements in the list.

You should provide both the signature and the implementation of the method. [6 marks]

(c) What is the complexity of your `printSearch` method from part (b) applied to a list of size n? Justify your answer. [3 marks]

END